

JEGYZŐKÖNYV

Mesterséges intelligencia és neurális hálózatok

Orosz Kristóf
EYZWG9

Sárospatak
2025. november 27.

Tartalomjegyzék

Bevezetés	3
A projekt genetikus algoritmusa	3
HTML fájlok összefoglalója	5
CSS fájl összefoglalója	25
Adatbázis létrehozása	27
PHP fájlok összefoglalója	28

Bevezetés

A mesterséges intelligencia és neurális hálózatok című tantárgyból egy elektromos rollereket tároló depók készletkiegyenlítését készítettem el. A projektem során a genetikus algoritmus elvét használtam az optimális útvonal meghatározására, vagyis arra, hogy a tehergépkocsi a lehető legrövidebb úton tudja elosztani vagy összegyűjteni a rollereket a különböző depók között. A projekt front-end részét HTML és CSS segítségével valósítottam meg, amely a felhasználói felületet biztosítja. A JavaScript felel az útvonaloptimalizálási folyamat és a genetikus algoritmus logikai megvalósításáért. A PHP-fájlok az adatbázis-kezelést látják el, amelyek a MariaDB-ben létrehozott „MI_Beadando” adatbázissal kommunikálnak.

A projekt genetikus algoritmusa

A genetikus algoritmust az útvonaloptimalizálás problémájának megoldására használtam, ahol a cél a depók közötti legrövidebb útvonal megtalálása. A program kezdetben véletlenszerűen generál több lehetséges útvonalat (populációt), majd ezeken iterál generációkon keresztül. A szelekció során a rövidebb útvonalak előnyt élveznek, a keresztezés során a jó megoldások kombinálódnak, a mutáció pedig véletlenszerűen módosítja az útvonalakat a változatosság érdekében. minden generáció után a legjobb megoldások maradnak életben, így a rendszer fokozatosan egyre hatékonyabb útvonalakat talál. A folyamat 200 generáción keresztül ismétlődik, és a végén a legjobb azaz a legrövidebb útvonal kerül ki eredményként.

```
<script>

    // Itt számítom ki a depók közötti euklideszi távolságot.
    function tavolsag(a, b) {
        return Math.sqrt((a.x - b.x) ** 2 + (a.y - b.y) ** 2);
    }

    // Ezzel a függvénygel kiszámítom egy adott útvonal teljes hosszát.
    function utHossz(ut, raktarak) {
        let osszesHossz = 0;
        for (let i = 0; i < ut.length - 1; i++) {
            const a = raktarak.find(d => d.nev === ut[i]);
            const b = raktarak.find(d => d.nev === ut[i + 1]);
            osszesHossz += tavolsag(a, b);
        }
        const start = raktarak.find(d => d.nev === ut[0]);
        const vege = raktarak.find(d => d.nev === ut[ut.length - 1]);
        osszesHossz += tavolsag(vege, start);
        return osszesHossz;
    }

    // Ezzel a függvénygel hozom létre az első populációt, amely véletlenszerű útvonalakat tartalmaz.
    // minden egyed egy depósorrend, vagyis a jármű egy lehetséges útvonala.
```

```

// Így indul el a keresés, több különböző megoldással egyszerre.
function letrehozPopulacio(raktarak, meret) {
    const populacio = [];
    for (let i = 0; i < meret; i++) {
        const egyed = [...raktarak.map(d => d.nev)];
        // A depók sorrendjét véletlenszerűen megkeverem (Fisher-Yates algoritmus)
        for (let j = egyed.length - 1; j > 0; j--) {
            const rand = Math.floor(Math.random() * (j + 1));
            [egyed[j], egyed[rand]] = [egyed[rand], egyed[j]];
        }
        populacio.push(egyed);
    }
    return populacio;
}

// Ezzel a függvényel rendelem a populációt az útvonalhossz szerint.
function rendezPopulacio(populacio, raktarak) {
    return populacio.sort((a, b) => utHossz(a, raktarak) - utHossz(b, raktarak));
}

// A keresztezés során két szülő útvonalból hozok létre egy új "gyermek" útvonalat.
// A céлом, hogy a jó megoldások tulajdonságai kombinálódjanak, és így egy még jobb
útvonal jöjjön létre.
function keresztez(sz1, sz2) {
    const start = Math.floor(Math.random() * sz1.length);
    const end = start + Math.floor(Math.random() * (sz1.length - start));
    const resz = sz1.slice(start, end);
    const gyerek = sz2.filter(g => !resz.includes(g));
    return [...resz, ...gyerek];
}

// A mutációt arra használom, hogy a genetikus keresés ne ragadjon be egy helyi
minimumba.
function mutacio(egyed, valoszinuseg) {
    const ut = [...egyed];
    if (Math.random() < valoszinuseg) {
        const i = Math.floor(Math.random() * ut.length);
        const j = Math.floor(Math.random() * ut.length);
        [ut[i], ut[j]] = [ut[j], ut[i]];
    }
    return ut;
}

// Ez a genetikus algoritmus fő függvénye, amely végrehajtja a teljes evolúciós
folyamatot.
function genetikusAlgoritmus(raktarak, generaciok = 200, populacioMeret = 60) {
    // Létrehozom a kezdeti populációt

```

```

let populacio = letrehozPopulacio(raktarak, populacioMeret);

// Itt zajlik az evolúciós ciklus - 200 generációon keresztül javítom a megoldásokat
for (let gen = 0; gen < generacio; gen++) {
    // A legjobb útvonalakat előresorolom
    populacio = rendezPopulacio(populacio, raktarak);
    const ujPop = [];

    // A két legjobb útvonalat közvetlenül továbbviszem
    ujPop.push(populacio[0], populacio[1]);
    while (ujPop.length < populacioMeret) {
        const sz1 = populacio[Math.floor(Math.random() * (populacioMeret / 2))];
        const sz2 = populacio[Math.floor(Math.random() * (populacioMeret / 2))];
        let gyerek = keresztez(sz1, sz2);
        gyerek = mutacio(gyerek, 0.2); // 20%-os mutációs eséllyel dolgozom
        ujPop.push(gyerek);
    }

    // A generáció végén az új populáció lesz az aktuális
    populacio = ujPop;
}

// Kiválasztom a legjobb (legrövidebb) útvonalat
const legjobb = rendezPopulacio(populacio, raktarak)[0];
const tav = utHossz(legjobb, raktarak).toFixed(2);

// A legjobb útvonalat és annak hosszát visszaadom
return { legjobb, tav };
}

```

HTML fájlok összefoglalója

Az index.html fájlt a projekt nyitóoldalaként készítettem el, amely az E-roller készletkiegyenlítő rendszer fő bemutató felülete. A felső menüsávban elhelyeztem a projekt címét, a saját azonosítómat, valamint két navigációs gombot, amelyek az útvonaltervezéshez és a rakodási naplóhoz vezetnek. A főoldal kétoszlopos elrendezésben jelenik meg: a bal oldalon a rendszer címe, a jobb oldalon pedig egy illusztrációs kép látható.

```

<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>E-roller készletkiegyenlítő</title>
    <link rel="stylesheet" href="menu.css">

```

```

<style>
    /* A fő tartalmat középre igazító konténer */
    .tartalom-kontener {
        flex: 1;
        display: flex;
        justify-content: center;
        align-items: center;
        gap: 40px;
        flex-wrap: wrap;
        text-align: center;
        padding: 20px;
    }

    /* Bal oldali szövegeszlop stílusa */
    .bal-oszlop {
        flex: 1;
        min-width: 280px;
        max-width: 400px;
    }

    /* Címsor méretének beállítása */
    .bal-oszlop h1 {
        font-size: 1.8em;
        margin-bottom: 20px;
    }

    /* Jobb oldali képoszlop stílusa */
    .jobb-oszlop {
        flex: 1;
        min-width: 280px;
        max-width: 400px;
    }

    /* A főoldali e-roller kép formázása */
    .roller-kep {
        max-width: 100%;
        height: auto;
        border-radius: 10px;
    }

    /* Linkek egységes megjelenése */
    a {
        color: white;
        text-decoration: none;
    }

    a:hover {
        color: white;
    }

```

```

        }

    /* Mobilnézethez: az elemek egymás alá kerülnek */
    @media (max-width: 768px) {
        .tartalom-kontener {
            flex-direction: column;
        }
    }

```

```
</head>

<body>
```

```
<!-- Felső menüsáv -->
```

```
<header class="menu">
    <div class="menu-bal">
```

```
        <a href="index.html">
            <b><i>E-roller készletkiegyenlítő</i></b>
            <span style="font-size: 14px;"><br>Orosz Kristóf – EYZWG9</span>
        </a>
    </div>
```

```
    <div class="menu-jobb">
        <a href="utvonaltervezes.html"><button class="menu-gomb"><i>ÚTVONALTERVEZÉS</i></button></a>
        <a href="rakodasinaplo.html"><button class="menu-gomb"><i>RAKODÁSI NAPLÓ</i></button></a>
    </div>
</header>
```

```
<main class="tartalom-kontener">
```

```
    <section class="bal-oszlop">
        <h1><b><i>Városi közösségi e-roller készletkiegyenlítő rendszer</i></b></h1>
    </section>
```

```
    <section class="jobb-oszlop">
        
    </section>
</main>
```

```
</body>
</html>
```

Az utvonaltervezes.html oldalt a rendszer adatbeviteli és előkészítő felületeként hoztam létre. Itt adhatóak meg a tehergépkosci jellemzői, például a kapacitás, a sebesség és az üzemanyag-fogyasztás, valamint kiválaszthatók a depók, amelyek adatai az adatbázisból töltődnek be. A JavaScript segítségével lehetőség van új depóadat-blokkok dinamikus hozzáadására is, így rugalmasan bővíthető a lista. A depók adatai a depok.php fájlon keresztül érkeznek a MariaDB adatbázisból, majd a felhasználó által kiválasztott elemeket a rendszer a localStorage-ban tárolja. A „Optimalizálás indítása” gombbal a program a terkep.html oldalra irányítja a felhasználót.

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Útvonaltervezés</title>
    <link rel="stylesheet" href="menu.css">

    <style>
        .urlap-kontener {
            max-width: 600px;
            margin: 40px auto;
            background-color: #f9ffff;
            border: 2px solid #90EE90;
            border-radius: 10px;
            padding: 25px;
            color: #006400;
        }

        fieldset {
            margin-bottom: 20px;
            border: 2px solid #90EE90;
            border-radius: 8px;
            padding: 15px;
        }

        legend {
            font-weight: bold;
        }

        label {
            display: block;
            font-weight: bold;
            margin-top: 10px;
        }

        input, select {
            display: block;
            width: 100%;
        }
    </style>

```

```

padding: 8px;
margin-top: 5px;
border: 1px solid #90EE90;
border-radius: 4px;
box-sizing: border-box;
}

.kuld-gomb {
display: block;
width: 100%;
background-color: #006400;
color: white;
border: none;
padding: 10px;
border-radius: 5px;
margin-top: 10px;
font-weight: bold;
cursor: pointer;
transition: 0.3s;
}

.kuld-gomb:hover {
background-color: #00a000;
}

</style>
</head>
<body>

<!-- Menü --&gt;
&lt;header class="menu"&gt;
&lt;div class="menu-bal"&gt;
&lt;a href="index.html" style="color:white;text-decoration:none;"&gt;
&lt;b&gt;&lt;i&gt;E-roller készletkiegyenlítő&lt;/i&gt;&lt;br&gt;
&lt;span style="font-size:14px;"&gt;Orosz Kristóf - EYZWG9&lt;/span&gt;&lt;/b&gt;
&lt;/a&gt;
&lt;/div&gt;

&lt;div class="menu-jobb"&gt;
&lt;a href="utvonaltervezes.html"&gt;&lt;button class="menu-gomb"&gt;&lt;i&gt;ÚTVONALTERVEZÉS&lt;/i&gt;&lt;/button&gt;&lt;/a&gt;
&lt;a href="rakodasinaplo.html"&gt;&lt;button class="menu-gomb"&gt;&lt;i&gt;RAKODÁSI NAPLÓ&lt;/i&gt;&lt;/button&gt;&lt;/a&gt;
&lt;/div&gt;
&lt;/header&gt;

<!-- Tartalom --&gt;
&lt;main class="urlap-kontener"&gt;
&lt;h1&gt;&lt;i&gt;Útvonaltervezés&lt;/i&gt;&lt;/h1&gt;
&lt;p&gt;Kérem, válasszon depót az útvonal megtervezéséhez!&lt;/p&gt;
</pre>

```

```

<fieldset id="depok">
    <legend>Depók adatai</legend>

    <div class="depo-blokk">
        <label>Depó neve:</label>
        <select class="depo-nev" onchange="betoltAdatok(this)">
            <option value="">-- Válassz depót --</option>
        </select>

        <label>X koordináta:</label>
        <input type="number" class="depo-x" readonly>

        <label>Y koordináta:</label>
        <input type="number" class="depo-y" readonly>

        <label>Aktuális készlet:</label>
        <input type="number" class="depo-aktualis" readonly>

        <label>Célkészlet:</label>
        <input type="number" class="depo-cel" readonly>
    </div>
</fieldset>

<button class="kuld-gomb" type="button" onclick="ujDepo()"+ Új depó
hozzáadása</button>
<button class="kuld-gomb" type="button" onclick="adatokatKuldes()">Optimalizálás
indítása</button>
</main>

<script>
    let depoAdatok = [];

    // Depók betöltése az adatbázisból
    function betoltDepok() {
        fetch("depok.php")
            .then(response => response.json())
            .then(data => {
                depoAdatok = data;
                feltoltDepoLista();
            })
            .catch(err => console.error("Hiba a depók betöltésekor:", err));
    }

    // Depók listájának frissítése
    function feltoltDepoLista() {
        document.querySelectorAll(".depo-nev").forEach(select => {
            const elozo = select.value;
            select.innerHTML = "<option value='>-- Válassz depót --</option>";
        });
    }
</script>

```

```

    depoAdatok.forEach(depo => {
      const opcio = document.createElement("option");
      opcio.value = depo.nev;
      opcio.textContent = depo.nev;
      if (depo.nev === elozo) opcio.selected = true;
      select.appendChild(opcio);
    });
  });
}

// Új depó blokk hozzáadása
function ujDepo() {
  const szulo = document.getElementById("depok");
  const uj = document.createElement("div");
  uj.classList.add("depo-blokk");

  uj.innerHTML =
    `Depó neve:
    <select class="depo-nev" onchange="betoltAdatok(this)">
      <option value="">-- Válassz depót --</option>
    </select>

    <label>X koordináta:</label>
    <input type="number" class="depo-x" readonly>

    <label>Y koordináta:</label>
    <input type="number" class="depo-y" readonly>

    <label>Aktuális készlet:</label>
    <input type="number" class="depo-aktualis" readonly>

    <label>Célkészlet:</label>
    <input type="number" class="depo-cel" readonly>
  `;
  szulo.appendChild(uj);
  feltoltDepoLista();
}

// Kiválasztott depó adatainak betöltése
function betoltAdatok(selectElem) {
  const talalat = depoAdatok.find(d => d.nev === selectElem.value);
  if (talalat) {
    const szulo = selectElem.closest(".depo-blokk");
    szulo.querySelector(".depo-x").value = talalat.x;
    szulo.querySelector(".depo-y").value = talalat.y;
    szulo.querySelector(".depo-aktualis").value = talalat.aktualis;
    szulo.querySelector(".depo-cel").value = talalat.cel;
  }
}

```

```

}

// Adatok mentése localStorage-ba
function adatokatKuldes() {
    const depok = [];

    document.querySelectorAll(".depo-blokk").forEach(d => {
        depok.push(d.querySelector(".depo-nev").value);
    });

    localStorage.setItem("kivalasztottDepok", JSON.stringify(depok));
    window.location.href = "terkep.html";
}

window.onload = betoltDepok;
</script>

</body>
</html>

```

A rakodasinaplo.html fájlt a depók aktuális készletének rögzítésére készítettem el. Ezen az oldalon a felhasználó kiválaszthatja a depó nevét, megadhatja az aktuális készletet, és a rendszer automatikusan beírja a lerakodás idejét. A depók listáját a rakodasinaplo.php fájlon keresztül töltöm be az adatbázisból. A JavaScript a fetch() metódus segítségével küldi el az adatokat a szerver felé, ahol azok mentésre kerülnek.

```

<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Rakodási napló</title>
    <link rel="stylesheet" href="menu.css">

    <style>
        .tartalomdoboz {
            flex: 1;
            display: flex;
            justify-content: center;
            align-items: center;
            gap: 30px;
            flex-wrap: wrap;
            padding: 40px 20px;
        }

        .urlapdoboz {
            flex: 1;

```

```
min-width: 300px;
max-width: 500px;
background-color: #f9ffff;
border: 2px solid #90EE90;
border-radius: 10px;
padding: 25px;
color: #006400;
}

fieldset {
    border: 2px solid #90EE90;
    border-radius: 8px;
    padding: 15px;
}

legend {
    font-weight: bold;
    color: #006400;
}

label {
    display: block;
    font-weight: bold;
    margin-top: 10px;
}

input,
select {
    display: block;
    width: 100%;
    padding: 8px;
    margin-top: 5px;
    border: 1px solid #90EE90;
    border-radius: 4px;
    box-sizing: border-box;
}

input:focus,
select:focus {
    outline: 2px solid #006400;
    background-color: #eaffea;
}

.mentes-gomb {
    display: block;
    width: 100%;
    background-color: #006400;
    color: white;
    border: none;
```

```

        padding: 10px;
        border-radius: 5px;
        margin-top: 10px;
        font-weight: bold;
        cursor: pointer;
        transition: 0.3s;
    }

.mentes-gomb:hover,
.mentes-gomb:focus {
    background-color: #00a000;
    outline: none;
}

@media (max-width: 768px) {
    .tartalomdoboz {
        flex-direction: column;
        text-align: center;
    }
}

```

</style>

</head>

<body>

<!-- Felső menüsáv a navigációhoz -->

<header class="menu">

<div class="menu-bal">

<i>E-roller készletkiegyenlítő</i>
0rosz

Kristóf - EYZWG9

</div>

<div class="menu-jobb">

<button class="menu-gomb" tabindex="1"><i>ÚTVONALTERVEZÉS</i></button>

<button class="menu-gomb" tabindex="2"><i>RAKODÁSI NAPLÓ</i></button>

</div>

</header>

<!-- A középre igazított űrlap -->

<div class="tartalomdoboz">

<main class="urlapdoboz">

<h1><i>Rakodási napló</i></h1>

<p>Kérlek, rögzítsd a lerakodási adatokat!</p>

<!-- Az űrlap a rakodási adatok rögzítésére szolgál -->

<form id="rakodasUrlap" onsubmit="mentes(); return false;">

```

<fieldset>
    <legend>Lerakodási adatok</legend>

    <!-- Depó neve lenyíló listából -->
    <label for="depoSelect">Depó neve:</label>
    <select id="depoSelect" name="depo" required tabindex="3">
        <option value="">-- Betöltés folyamatban... --</option>
    </select>

    <!-- Aktuális készlet beviteli mező -->
    <label for="aktualis">Aktuális készlet (db):</label>
    <input type="number" id="aktualis" name="aktualis" placeholder="Pl. 32" required
tabindex="4">

    <!-- A lerakodás időpontja automatikusan kitöltve -->
    <label for="ido">Lerakodás ideje:</label>
    <input type="datetime-local" id="ido" name="ido" readonly required tabindex="5">
</fieldset>

    <!-- Mentés gomb -->
    <button class="mentes-gomb" type="submit" tabindex="6">Adatok mentése</button>
</form>
</main>
</div>

<script>
    // A depók neveit a szerverről tölti be
    function betoltDepok() {
        fetch("rakodasinaplo.php")
            .then(res => res.json())
            .then(adatok => {
                const select = document.getElementById("depoSelect");
                select.innerHTML = "<option value='>-- Válassz depót --</option>";

                if (adatok.error) {
                    const option = document.createElement("option");
                    option.textContent = adatok.error;
                    select.appendChild(option);
                } else {
                    adatok.forEach(depo => {
                        const opcio = document.createElement("option");
                        opcio.value = depo.nev;
                        opcio.textContent = depo.nev;
                        select.appendChild(opcio);
                    });
                }
            })
            .catch(err => {
                console.error("Hiba a depók betöltésekor:", err);
            }
        );
    }
</script>

```

```

        document.getElementById("depoSelect").innerHTML =
            "<option>Hiba a betöltés során</option>";
    });
}

// Az aktuális idő automatikus beállítása
function frissitIdo() {
    const idoMezo = document.getElementById("ido");
    const most = new Date();
    const helyi = new Date(most.getTime() - most.getTimezoneOffset() * 60000)
        .toISOString()
        .slice(0, 16);
    idoMezo.value = helyi;
}

// Az űrlapadatok mentése a szerver felé
function mentes() {
    const depo = document.getElementById("depoSelect").value;
    const aktualis = document.getElementById("aktualis").value;
    const ido = document.getElementById("ido").value;

    if (!depo || !aktualis || !ido) {
        alert("Kérlek, tölts ki minden mezőt!");
        return;
    }

    fetch("rakodasinaplo.php", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ depo, aktualis, ido })
    })
        .then(res => res.json())
        .then(valasz => {
            if (valasz.success) {
                alert(valasz.success);
                document.getElementById("rakodasUrlap").reset();
                frissitIdo();
            } else {
                alert(valasz.error || "Ismeretlen hiba történt.");
            }
        })
        .catch(err => {
            console.error("Hiba a mentés során:", err);
            alert("Hálózati hiba történt a mentés közben!");
        });
}

// Billentyűzetkezelés: Enter gombbal is menthető az űrlap
document.addEventListener("keydown", function (e) {

```

```

        if (e.key === "Enter") {
            e.preventDefault();
            mentes();
        }
    });

// Az oldal betöltésekor automatikusan betölti a depókat és az időt
window.onload = function () {
    betoltDepok();
    frissitIdo();
    setInterval(frissitIdo, 1000 * 30);
};

</script>

</body>
</html>

```

A terkep.html fájlt a depók térképi megjelenítésére és az útvonaloptimalizálás működésének vizuális szemléltetésére készítettem el. A kiválasztott depók a rácsos térképen jelennek meg, a helyüket az adatbázisban tárolt koordináták alapján határozzom meg. A genetikus algoritmust JavaScriptben valósítottam meg, amely több generációt keresztül keresi a legrövidebb lehetséges útvonalat a depók között. Az optimalizálás eredményeként megjelenítem a legjobb útvonalat, valamint a hozzá tartozó távolságot.

```

<!DOCTYPE html>
<html lang="hu">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Rácsos térkép - E-roller készletkiegyenlítő</title>
    <link rel="stylesheet" href="menu.css">

    <style>
        /* A térképet megjelenítő 20x20-as rács stílusa */
        .racs-kontener {
            display: grid;
            grid-template-columns: repeat(20, 30px);
            grid-template-rows: repeat(20, 30px);
            gap: 2px;
            margin: 40px auto;
            background-color: #d9fcf9;
            border: 2px solid #006400;
            border-radius: 10px;
            padding: 5px;
            position: relative;
            width: fit-content;
        }
    </style>

```

```

    z-index: 1;
}

/* Egy-egy rácselem (cella) megjelenítése */
.cella {
    width: 30px;
    height: 30px;
    background-color: #ffffff;
    border: 1px solid #c2e5c2;
    border-radius: 4px;
    position: relative;
}

/* A depókat jelölő zöld körök formázása */
.raktar {
    background-color: #006400;
    border-radius: 50%;
    width: 22px;
    height: 22px;
    position: absolute;
    top: 4px;
    left: 4px;
    cursor: pointer;
    transition: transform 0.2s;
    z-index: 2;
}

/* Ha az egér a depó fölé kerül, nagyít és színt vált */
.raktar:hover {
    transform: scale(1.2);
    background-color: #00a000;
}

/* A kor (tooltip), amely a depó adatait mutatja */
.kor {
    position: absolute;
    background-color: rgba(0, 100, 0, 0.9);
    color: white;
    padding: 4px 8px;
    border-radius: 5px;
    font-size: 12px;
    white-space: nowrap;
    transform: translate(-50%, -120%);
    pointer-events: none;
    opacity: 0;
    transition: opacity 0.2s;
}

/* kor megjelenése egér fölé vitelekor */

```

```

.raktar:hover .kor {
    opacity: 1;
}

/* Az optimalizálás indítására szolgáló gomb */
.optimalizalas-gomb {
    margin: 15px auto;
    background-color: #006400;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-weight: bold;
    font-size: 1em;
    transition: 0.3s;
    display: block;
    width: fit-content;
}

.optimalizalas-gomb:hover {
    background-color: #00a000;
}

/* Az eredmény megjelenítsi doboz */
#utvonalEredmeny {
    margin: 20px auto;
    background: #f0ffff;
    border: 2px solid #90EE90;
    border-radius: 10px;
    padding: 15px;
    max-width: 600px;
    font-weight: bold;
    color: #004d00;
    white-space: pre-wrap;
    display: none;
}

/* Mobilbarát megjelenés */
@media (max-width: 768px) {
    .racs-kontener {
        grid-template-columns: repeat(10, 25px);
        grid-template-rows: repeat(10, 25px);
    }
}
</style>
</head>

<body>

```

```


<header class="menu">
  <div class="menu-bal">
    <a href="index.html" style="text-decoration: none; color: white;">
      <b><i>E-roller készletkiegyenlítő</i><br>
        <span style="font-size:12px;">Orosz Kristóf - EYZWG9</span></b>
    </a>
  </div>
  <div class="menu-jobb">
    <a href="utvonaltervezes.html"><button class="menu-gomb"><i>ÚTVONALTERVEZÉS</i></button></a>
    <a href="rakodasinaplo.html"><button class="menu-gomb"><i>RAKODÁSI NAPLÓ</i></button></a>
  </div>
</header>


<div id="racs" class="racs-kontener"></div>


<button class="optimalizalas-gomb">Optimalizálás indítása</button>


<div id="utvonaleredmeny"></div>

<script>
  // A kiválasztott depókat a localStorage-ból olvassa be
  const kivalasztottRaktarak = JSON.parse(localStorage.getItem("kivalasztottDepok") || "[]");

  // Ha nincs depó kiválasztva, visszairányít az útvonaltervezéshez
  if (kivalasztottRaktarak.length === 0) {
    alert("Nincsenek kiválasztott depók. Kérlek, térd vissza az útvonaltervezéshez!");
    window.location.href = "utvonaltervezes.html";
  }

  // A depók adatainak betöltése a PHP fájlból
  fetch("depok.php")
    .then(valasz => valasz.json())
    .then(adataok => {
      const racs = document.getElementById("racs");

      // 20x20-as rács létrehozása
      for (let i = 0; i < 400; i++) {
        const cella = document.createElement("div");
        cella.classList.add("cella");
        racs.appendChild(cella);
      }
    })
</script>

```

```

// Kiválasztott raktárak megjelenítése a rácson
adatok.forEach(raktar => {
    if (kivalasztottRaktarak.includes(raktar.nev)) {
        const x = parseInt(raktar.x);
        const y = parseInt(raktar.y);
        const index = y * 20 + x;
        const cella = racs.children[index];
        if (cella) {
            const raktarElem = document.createElement("div");
            raktarElem.classList.add("raktar");
            const kor = document.createElement("div");
            kor.classList.add("kor");
            kor.textContent = `${raktar.nev}\nKészlet: ${raktar.aktualis}/${raktar.cel}`;
            raktarElem.appendChild(kor);
            cella.appendChild(raktarElem);
        }
    }
});
});

</script>

// A depók közti távolság kiszámítása
function tavolsag(a, b) {
    return Math.sqrt((a.x - b.x) ** 2 + (a.y - b.y) ** 2);
}

// Egy útvonal teljes hosszának kiszámítása
function utHossz(ut, raktarak) {
    let osszesHossz = 0;
    for (let i = 0; i < ut.length - 1; i++) {
        const a = raktarak.find(d => d.nev === ut[i]);
        const b = raktarak.find(d => d.nev === ut[i + 1]);
        osszesHossz += tavolsag(a, b);
    }
    const start = raktarak.find(d => d.nev === ut[0]);
    const vege = raktarak.find(d => d.nev === ut[ut.length - 1]);
    osszesHossz += tavolsag(vege, start);
    return osszesHossz;
}

// Kezdeti populáció létrehozása (véletlen sorrendű utak)
function letrehozPopulacio(raktarak, meret) {
    const populacio = [];
    for (let i = 0; i < meret; i++) {
        const egyed = [...raktarak.map(d => d.nev)];
        for (let j = egyed.length - 1; j > 0; j--) {
            const rand = Math.floor(Math.random() * (j + 1));

```

```

        [egyed[j], egyed[rand]] = [egyed[rand], egyed[j]];
    }
    populacio.push(egyed);
}
return populacio;
}

// Populáció rendezése útvonalhossz szerint
function rendezPopulacio(populacio, raktarak) {
    return populacio.sort((a, b) => utHossz(a, raktarak) - utHossz(b, raktarak));
}

// Két útvonal keresztezése
function keresztek(sz1, sz2) {
    const start = Math.floor(Math.random() * sz1.length);
    const end = start + Math.floor(Math.random() * (sz1.length - start));
    const resz = sz1.slice(start, end);
    const gyerek = sz2.filter(g => !resz.includes(g));
    return [...resz, ...gyerek];
}

// Mutáció véletlenszerű csere formájában
function mutacio(egyed, valoszinuseg) {
    const ut = [...egyed];
    if (Math.random() < valoszinuseg) {
        const i = Math.floor(Math.random() * ut.length);
        const j = Math.floor(Math.random() * ut.length);
        [ut[i], ut[j]] = [ut[j], ut[i]];
    }
    return ut;
}

// A genetikus algoritmus fő függvénye
function genetikusAlgoritmus(raktarak, generaciok = 200, populacioMeret = 60) {
    let populacio = letrehozPopulacio(raktarak, populacioMeret);
    for (let gen = 0; gen < generaciok; gen++) {
        populacio = rendezPopulacio(populacio, raktarak);
        const ujPop = [];
        ujPop.push(populacio[0], populacio[1]);
        while (ujPop.length < populacioMeret) {
            const sz1 = populacio[Math.floor(Math.random() * (populacioMeret / 2))];
            const sz2 = populacio[Math.floor(Math.random() * (populacioMeret / 2))];
            let gyerek = keresztek(sz1, sz2);
            gyerek = mutacio(gyerek, 0.2);
            ujPop.push(gyerek);
        }
        populacio = ujPop;
    }
    const legjobb = rendezPopulacio(populacio, raktarak)[0];
}

```

```

        const tav = utHossz(legjobb, raktarak).toFixed(2);
        return { legjobb, tav };
    }

// A legjobb útvonal kirajzolása vonalakkal
function utvonalRajzolas(legjobbUtvonal, raktarak) {
    const racs = document.getElementById("racs");
    let svg = document.getElementById("utvonal-svg");
    if (!svg) {
        svg = document.createElementNS("http://www.w3.org/2000/svg", "svg");
        svg.setAttribute("id", "utvonal-svg");
        svg.style.position = "absolute";
        svg.style.top = racs.offsetTop + "px";
        svg.style.left = racs.offsetLeft + "px";
        svg.style.width = racs.offsetWidth + "px";
        svg.style.height = racs.offsetHeight + "px";
        svg.style.pointerEvents = "none";
        document.body.appendChild(svg);
    }

    const pontok = legjobbUtvonal.map(n => raktarak.find(d => d.nev === n));
    const cellaMeret = 32;
    const offsetX = racs.offsetLeft + 15;
    const offsetY = racs.offsetTop + 15;

    pontok.forEach((p, i) => {
        if (i < pontok.length - 1) {
            const a = pontok[i];
            const b = pontok[i + 1];
            const x1 = a.x * cellaMeret + offsetX;
            const y1 = a.y * cellaMeret + offsetY;
            const x2 = b.x * cellaMeret + offsetX;
            const y2 = b.y * cellaMeret + offsetY;

            const vonal = document.createElementNS("http://www.w3.org/2000/svg", "line");
            vonal.setAttribute("x1", x1);
            vonal.setAttribute("y1", y1);
            vonal.setAttribute("x2", x2);
            vonal.setAttribute("y2", y2);
            vonal.setAttribute("stroke", "#008000");
            vonal.setAttribute("stroke-width", "3");
            vonal.style.opacity = 0;
            svg.appendChild(vonal);
            setTimeout(() => {
                vonal.style.transition = "opacity 0.5s ease";
                vonal.style.opacity = 1;
            }, i * 400);
        }
    });
}

```

```

}

// A "Optimalizálás indítása" gomb eseménykezelője
document.querySelector(".optimalizalas-gomb").addEventListener("click", () => {
  fetch("depok.php")
    .then(res => res.json())
    .then(adatok => {
      const kivalasztottRaktarak = JSON.parse(localStorage.getItem("kivalasztottDepok"))
      || "[]";
      const aktivRaktarak = adatok.filter(d => kivalasztottRaktarak.includes(d.nev));
      const { legjobb, tav } = genetikusAlgoritmus(aktivRaktarak, 200);

      // Frissíti a készletet: a célkészlet felé mozdítja az értékeket
      const frissitesek = aktivRaktarak.map(d => {
        const kulonbseg = (d.cel - d.aktualis);
        const uj_aktualis = d.aktualis + Math.round(kulonbseg * 0.5);
        return { nev: d.nev, uj_aktualis };
      });

      // Frissítések küldése a PHP felé
      fetch("keszletfrissites.php", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ frissitesek })
      });

      // Kiírja a legjobb útvonalat és a távolságot
      const eredményDiv = document.getElementById("utvonalEredmeny");
      eredményDiv.style.display = "block";
      eredményDiv.innerHTML = `<b>Legjobb útvonal:</b><br>${legjobb.join(" →")}<br><br><b>Teljes távolság:</b> ${tav} egység`;

      utvonalRajzolas(legjobb, aktivRaktarak);
    });
  });
</script>
</body>
</html>

```

CSS fájl összefoglalója

Ezzel a CSS-kódrészlettel a weboldal kinézetét és elrendezését határoztam meg.

```

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background-color: #90EE90;
  color: #006400;
  height: 100vh;
}

```

```
        display: flex;
        flex-direction: column;
    }

.menu {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: green;
    color: white;
    padding: 15px 30px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}

.menu-bal {
    font-size: 1.2em;
    font-weight: bold;
}

.menu-gomb {
    background-color: transparent;
    border: 2px solid white;
    color: white;
    padding: 8px 16px;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    transition: 0.3s;
}

.menu-gomb:hover {
    background-color: white;
    color: #006400;
}

.tartalom-kontener {
    flex: 1;
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 40px;
    flex-wrap: wrap;
    text-align: center;
    padding: 20px;
}

.bal-oszlop {
```

```

        flex: 1;
        min-width: 280px;
        max-width: 400px;
    }

.bal-oszlop h1 {
    font-size: 1.8em;
    margin-bottom: 20px;
}

.jobb-oszlop {
    flex: 1;
    min-width: 280px;
    max-width: 400px;
}

@media (max-width: 768px) {
    .menu {
        flex-direction: column;
        text-align: center;
    }

    .menu-jobb {
        margin-top: 10px;
    }
}
}

```

Adatbázis létrehozása

Ebben a részben létrehoztam a projekt adatbázisát MariaDB-ben.

```
MariaDB [(none)]> CREATE DATABASE MI_beadando;
Query OK, 1 row affected (0.003 sec)
```

Létrehozom a depok című táblát.

```
MariaDB [(none)]> use MI_Beadando;
Database changed
MariaDB [MI_Beadando]> CREATE TABLE depok (
    -> id INT AUTO_INCREMENT PRIMARY KEY,
    -> nev VARCHAR(100) NOT NULL,
    -> x INT NOT NULL,
    -> y INT NOT NULL,
    -> aktualis INT NOT NULL,
    -> cel INT NOT NULL
    -> );
Query OK, 0 rows affected (0.024 sec)
```

Feltöltöm sárospataki utcanevekkel.

```

MariaDB [MI_Beadando]> INSERT INTO depok (nev, x, y, aktualis, cel) VALUES
-> ('Rákóczi út, 3950 Sárospatak', 0, 0, 26, 30),
-> ('Comenius utca, 3950 Sárospatak', 3, 5, 14, 22),
-> ('Bartók Béla utca, 3950 Sárospatak', 6, 3, 18, 25),
-> ('Árpád utca, 3950 Sárospatak', 8, 6, 12, 20),
-> ('Szent Erzsébet utca, 3950 Sárospatak', 1, 4, 28, 24),
-> ('Kazinczy Ferenc utca, 3950 Sárospatak', 9, 2, 20, 27),
-> ('Dózsa György utca, 3950 Sárospatak', 5, 7, 15, 18),
-> ('Eötvös József utca, 3950 Sárospatak', 4, 2, 19, 23),
-> ('József Attila utca, 3950 Sárospatak', 2, 8, 10, 17),
-> ('Hunyadi János utca, 3950 Sárospatak', 7, 9, 22, 26);
Query OK, 10 rows affected (0.011 sec)
Records: 10  Duplicates: 0  Warnings: 0

```

Létrehozom a rakodasinaplo című táblát amire a készletek frissítése miatt van szükségem.

```

MariaDB [MI_Beadando]> CREATE TABLE rakodasinaplo (
->     id INT AUTO_INCREMENT PRIMARY KEY,
->     depo_nev VARCHAR(255) NOT NULL,
->     aktualis INT NOT NULL,
->     ido DATETIME NOT NULL
-> );
Query OK, 0 rows affected (0.015 sec)

```

PHP fájlok összefoglalója

Ebben a PHP-fájlban hozom létre az adatbázis-kapcsolatot a MI_Beadando nevű adatbázissal.

```

<?php
// Az adatbázis-kapcsolat beállításához szükséges adatok megadása.
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "MI_Beadando";

// Létrehozom a kapcsolatot a MySQL adatbázissal a mysqli osztály
// segítségével.
$conn = new mysqli($servername, $username, $password, $dbname);

// Beállítom a karakterkódolást UTF-8-ra, hogy a magyar ékezetes karakterek is
// helyesen jelenjenek meg.
$conn->set_charset("utf8mb4");

// Ellenőrzöm, hogy sikerült-e a kapcsolat.
if ($conn->connect_error) {
    die("Adatbázis kapcsolat sikertelen: " . $conn->connect_error);
}
?>

```

Ebben a PHP-fájlban a **depok** tábla adatait olvasom ki az adatbázisból, majd JSON formátumban adom vissza a kliensoldal számára.

```
<?php
header('Content-Type: application/json; charset=utf-8');

// Betöltöm az adatbázis-kapcsolatot a kapcsolat.php fájlból.
require_once 'kapcsolat.php';

// Lekérdezem az összes depót az adatbázisból a szükséges mezőkkel.
$sql = "SELECT id, nev, x, y, aktualis, cel FROM depok";
$result = $conn->query($sql);

// Egy üres tömböt hozok létre, amibe a lekérdezett adatokat fogom eltárolni.
$depok = [];

// Ha a lekérdezés sikeres és van eredmény, bejárom a sorokat és beleteszem a tömbbe.
if ($result && $result->num_rows > 0) {
    while ($row = $result->fetch_assoc()) {
        $depok[] = $row;
    }
    // A PHP tömböt JSON formátumba alakítom és visszaküldöm a kliensnek.
    echo json_encode($depok, JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
} else {
    // Ha nincs adat, hibaüzenetet küldök JSON formátumban.
    echo json_encode(["error" => "Nem található adat a depok táblában."]);
}

// A kapcsolatot lezárom, hogy ne foglalja tovább az erőforrásokat.
$conn->close();
?>
```

Ebben a PHP-fájlban a **depók adatainak lekérdezését és frissítését** valósítom meg kétféle http metódus alapján. GET kérés esetén lekérem a depók nevét és azonosítóját, amit a front-end a lenyíló listához használ. POST kérés esetén a felhasználó által megadott értékekkel frissítem a depók aktuális készletét az adatbázisban.

```
<?php
header('Content-Type: application/json; charset=utf-8');

// Betöltöm az adatbázis kapcsolatot tartalmazó fájlt.
require_once 'kapcsolat.php';

// Lekérem a HTTP metódust hogy tudjam, milyen műveletet kell végrehajtani.
$method = $_SERVER['REQUEST_METHOD'];

// Ha GET metódussal érkezik a kérés, akkor lekérem a depók listáját az adatbázisból.
```

```

if ($method === 'GET') {
    // A depók nevét és azonosítóját kérem le, hogy a lenyíló lista feltölthető legyen.
    $sql = "SELECT id, nev FROM depok";
    $result = $conn->query($sql);

    $depok = [];

    // Ha van találat, akkor minden sort beolvasok és egy tömbbe gyűjtök.
    if ($result && $result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            $depok[] = $row;
        }
        // A találatokat JSON formátumban visszaküldöm a kliensnek.
        echo json_encode($depok, JSON_UNESCAPED_UNICODE | JSON_PRETTY_PRINT);
    } else {
        // Ha nincs adat, akkor hibaüzenetet küldök.
        echo json_encode(["error" => "Nincs elérhető depó az adatbázisban."]);
    }
}

// Ha POST metódus érkezik, akkor az aktuális készletet frissítem a depók táblában.
elseif ($method === 'POST') {
    // A beérkező nyers JSON adat beolvasása.
    $input = file_get_contents("php://input");

    $data = json_decode($input, true);

    // Ellenőrzöm, hogy a szükséges adatok megvannak-e.
    if (!isset($data['depo'], $data['aktuális'], $data['ido'])) {
        echo json_encode(["error" => "Hiányzó adatok a kérésben."]);
        exit;
    }

    // Az adatok biztonságos feldolgozása az SQL injection elkerülése érdekében.
    $depo = $conn->real_escape_string($data['depo']);
    $aktuális = intval($data['aktuális']);
    $ido = $conn->real_escape_string($data['ido']);

    // A depó aktuális készletének frissítése az adatbázisban.
    $sql = "UPDATE depok SET aktuális = $aktuális WHERE nev = '$depo'";

    // Ha sikeres a frissítés, sikerüzenetet küldök.
    if ($conn->query($sql) === TRUE) {
        echo json_encode(["success" => "A(z) $depo depó aktuális készlete frissítve:
$aktuális db."]);
    } else {
        // Ha hiba történik, hibaüzenetet küldök.
        echo json_encode(["error" => "Hiba a frissítés során: " . $conn->error]);
    }
}

```

```
}
```

```
// Az adatbázis kapcsolat lezárása.
```

```
$conn->close();
```

```
?>
```